**Strings**

**Important Points to remember:**

* String represents a sequence of characters.
* String is a final class (cannot be inherited) which is present in **java.lang.package**.
* String objects are “immutable”, i.e. once the object is created then the value cannot be changed.
* The [==] operator always checks for the object reference rather [.equals] operator checks for the object content equality.
* The equals’ method is defined in the parent most class called object and this method is overridden in string class.
* String is a class in java.lang.package. All classes in java are also considered as data types, so we can take string as a data type also.
* We can call class is also a ‘user-defined’ data type. This is because user can create a class.

**String methods with example:**

|  |
| --- |
| **Program:** |
| **package** com.sireesh;  **public** **class** StringMethods {  **public** **static** **void** main(String[] args) {  String s1="SIREESH";  String s2="sireesh";  String s4="sireesh";    String s3="Sudhir";      System.*out*.println("To lower Case:::"+s1.toLowerCase());    System.*out*.println("To Upper Case:::"+s2.toUpperCase());      System.*out*.println("Equality By Igoring the case::::"+s1.equalsIgnoreCase(s2));      System.*out*.println("Equality:::"+s2.equals(s4));    System.*out*.println("String Length:::"+s1.length());    System.*out*.println("Perticular Character:::"+s1.charAt(6));    System.*out*.println("Returns negative if s1<s2,positive if s1>s2 and zero if s1=s2:::"+s4.compareTo(s2));    System.*out*.println("Returns negative if s1<s2,positive if s1>s2 and zero if s1=s2:::"+s1.compareToIgnoreCase(s2));    System.*out*.println("String Concatination:::"+s1.concat(s2));    System.*out*.println("Substring:::"+s1.substring(1));    System.*out*.println("Substring::::"+s1.substring(2, 4));    System.*out*.println("Index of a Character in a string:::"+s1.indexOf("I"));      System.*out*.println("Replace all appearences of i with s:::"+s1.replace("I", "S"));    String s=" pradeep";      System.*out*.println("removes white spaces:::"+s.trim());  }  } |

|  |
| --- |
| **Output:** |
| To lower Case:::sireesh  To Upper Case:::SIREESH  Equality By Igoring the case::::true  Equality:::true  String Length:::7  Perticular Character:::H  Returns negative if s1<s2,positive if s1>s2 and zero if s1=s2:::0  Returns negative if s1<s2,positive if s1>s2 and zero if s1=s2:::0  String Concatination:::SIREESHsireesh  Substring:::IREESH  Substring::::RE  Index of a Character in a string:::1  Replace all appearences of i with s:::SSREESH  removes white spaces:::pradeep |

**String Buffer and String Builder:**

**Important Points to remember (String Buffer):**

* String Buffer class objects are ‘mutable’, so their contents can be modified.
* The methods that directly manipulate data of the object are available in String Buffer class.
* StringBuffer class is synchronized (**Thread Safe**), when the programmer wants to use several threads, he should use StringBuffer as it gives reliable results.

**StringBuffer methods with example:**

|  |
| --- |
| **Program:** |
| **package** com.sireesh;  **public** **class** StringBufferMethods {    **public** **static** **void** main(String[] args) {  StringBuffer sb=**new** StringBuffer("Object Oriented");    System.*out*.println("String buffer without append:::"+sb);    //appending a string at the end  sb.append(" langauage");  System.*out*.println("Appends string at the end:::"+sb);      sb.setCharAt(6,'-');  System.*out*.println("Inserts the string s2 at the position n of string s1:::"+sb);    //Insering a string in the middle  StringBuffer s=**new** StringBuffer("Inteligent person");  s.insert(11, "young");  System.*out*.println("Insering a string in the middle:::"+s);      //string reverse  s.reverse();  System.*out*.println("Reverse of a string:::"+s);      }  } |
| **Output:** |
| String buffer without append:::Object Oriented  Appends string at the end:::Object Oriented langauage  Inserts the string s2 at the position n of string s1:::Object-Oriented langauage  Insering a string in the middle:::Inteligent young person  Reverse of a string:::nosrep gnuoy tnegiletnI |

**Important Points to remember (String Builder):**

* String Builder class objects are ‘mutable’, so their contents can be modified.
* StringBuilder class is not synchronized, when the programmer wants to use only one thread, he should use StringBuilder as it gives reliable results.
* StringBuilder methods are same as StringBuffer methods.

**Interview Questions:**

**1. Is String is a class or data type?**

**Ans: String is a class in Java.lang package, but in java, all classes are also considered as data types, So we can take string as data type also.**

**2. Can we call a class as a data type?**

**Ans: Yeas, a class is also called ‘user-defined’ data type. This is because a user can create a class.**

**3. What is the difference between == and equals () while comparing Strings? Which one is reliable?**

**Ans: The [==] operator always checks for reference rather [.equals ()] operator checks for the object content equality.**

**4. What is a string constant pool?**

**Ans: String constant pool is a separate block of memory where the string objects are held by JVM.**

**If a string object is created directly, then it is stored in string constant pool.**

**5. What do you mean by StringBuffer?**

**Ans: StringBuffer represents strings in such a way that their data can be modified, it means StringBuffer objects are mutable (can be changed)**

**6. What is the difference between String and StringBuffer classes?**

**Ans: String class objects are immutable and hence their contents cannot be modified rather StringBuffer class objects are mutable and hence their contents can be modified.**

**7. Are there any other classes whose objects are immutable?**

**Ans: Yes, Classes like Character, Byte, Integer,,,, called ‘Wrapper Classes’ are immutable and Classes like Class, BigInteger, BigDecimal, are also immutable.**

**8. What do you mean by StringBuilder?**

**Ans: StringBuilder represents a string in such a way that their data can be modified, It means StringBuilder objects are mutable (can be changed ) as like in StringBuffer.**

**9. What is the Difference between StringBuffer and StringBuilder?**

|  |  |
| --- | --- |
| **StringBuffer** | **StringBuilder** |
| **StringBuffer class is synchronized.** | **StringBuilder class is not synchronized.** |
| **When the programmer wants to use several threads, he should use StringBuffer as it gives Reliable results.** | **When the programmer wants to use only one thread, StringBuilder is preferred, as it improves execution time.** |

**Interview Programs:**

1. Write a program for testing whether it is palindrome or not.

|  |
| --- |
| Program: |
| **package** com.sireesh;  **public** **class** Palindrome {  **public** **static** **void** main(String[] args) {  String str="152";    String temp=str;    StringBuffer sb=**new** StringBuffer(str);  sb.reverse();    //convert the stringbuffer to a string  str=sb.toString();  **if**(temp.equalsIgnoreCase(str))  {  System.*out*.println(temp+ " is palindrome");  }  **else**  {  System.*out*.println(temp+ " is not a palindrome");  }    }  } |
| **Output:** |
| 152 is not a palindrome |

2. Write a program to reverse a string without using in built methods.

|  |
| --- |
| Program: |
| **package** com.sireesh;  **public** **class** StringReverse {  **public** **static** **void** main(String[] args) {  String s = "Sireesh";  **char**[] temp = s.toCharArray();  **int** l = s.length() - 2;  **int** li = s.length() - 1;  **for** (**int** i = 0; i < l; i++) {  **char** x = temp[i];  temp[i] = temp[li - i];  temp[li - i] = x;  }  System.*out*.println(s);  System.*out*.println(**new** String(temp));  }    } |
| **Output:** |
| Sireesh  hsreeiS |